

REMARKS

Claims 1-16 are pending.

In the Office Action, claims 1-16 were rejected under 35 U.S.C. § 103(a) as obvious in view of Long (U.S. Patent No. 6,691,307) and Seshadri (U.S. Patent No. 6,658,421).

The rejection of claims 1-16 is not supported by the cited art and should be withdrawn for the reasons below.

Claim 1, by way of example, recites:

A computer system comprising:  
a preloader arranged to,  
determine whether a bytecode makes an active  
reference to a class which requires an execution of a static  
initializer,  
determine if the class has a superclass which  
requires the execution of the static initializer, wherein the  
preloader produces a source file;  
a compiler coupled to the preloader arranged to accept the  
source file as input and produce an object file . . . (Emphasis  
Added).

The computer system defined by claim 1 includes the feature of a preloader. The preloader itself provides several features. The preloader is arranged to make certain determinations, and the preloader produces a source file. A compiler coupled to the preloader is arranged to accept the source file as input and produce an object file. The preloader, by making the determinations recited in claim 1, allows for reduction of the number of runtime checks performed during execution of the object file.

The Office Action states, on page 2, that "Long does not explicitly disclose a preloader arranged to, determine whether a bytecode makes an active reference to a class which requires an execution of a static initializer, determine if the class has a superclass which requires the

execution of the static initializer, wherein the preloader produces a source file.” Applicant agrees with this statement.

The Office Action further states, on pages 2-3, that “Seshadri in an analogous art teaches ‘a preloader arranged to, determine whether a bytecode (E.g. see col. 4:56, invokestatic) makes an active reference to a class which requires an execution of a static initializer, determine if the class has a superclass (E.g. see col. 4:66 and col. 5:5) which requires the execution of the static initializer, wherein the preloader produces a source file.’ (E.g. see TABLE 2 at col. 12 and see col. 4:54 to col. 5:5).” As explained below, the above-cited portions of Seshadri do not support these assertions in the Office Action. Seshadri would fail to cure the deficiencies of Long even if Seshadri could somehow be combined with Long. Therefore, the rejection of claim 1 as obvious in view of Long and Seshadri should be withdrawn.

Seshadri does not apply to the computer system of claim 1. This is because Seshadri fails to disclose or suggest a preloader as defined by claim 1, including all of the features of the preloader as described above. That is, Seshadri fails to disclose or suggest: (1) a preloader arranged to, (2) determine whether a bytecode makes an active reference to a class which requires an execution of a static initializer, (3) determine if the class has a superclass which requires the execution of the static initializer, and (4) wherein the preloader produces a source file.

Instead, Seshadri teaches techniques for detecting binary compatibility in compiled object code. (col. 3, lines 63-64). According to Seshadri, characterizing indicia, e.g., a method block table signature, is encoded into class metadata during compilation. (col. 4, lines 2-5, 22-26). Seshadri explains that the method block table signature is encoded when compiling an “invokestatic bytecode.” (col. 4, lines 55-56). Seshadri goes on to explain the general procedure for loading classes in a Java virtual machine (col. 12, table 2).

Nowhere does Seshadri disclose or suggest a preloader, much less the features of the preloader defined by claim 1. This is because Seshadri's techniques are performed during compilation. Following the teachings of Seshadri, the source file would have to be generated and output to the compiler for compiling, before any processing would begin. Then the compiler or other apparatus cooperating with the compiler would have to perform the encoding during compilation of the source file to produce an object file. Therefore, the encoding techniques described by Seshadri could not be performed by "a preloader ... wherein the preloader produces a source file," as recited in claim 1.

In addition, Seshadri fails to disclose or suggest the features of the preloader defined by claim 1, namely determining whether a bytecode makes an active reference to a class, and/or if the class has a superclass. Instead, as demonstrated by the above-quoted portions, Seshadri's teachings relate to techniques for encoding signatures and characterizing indicia. Neither of the determinations, made by the preloader as defined by claim 1, are disclosed nor suggested by Seshadri. Seshadri only teaches encoding signatures and characterizing indicia without "determin[ing] whether a bytecode makes an active reference to a class which requires an execution of a static initializer," or "determin[ing] if the class has a superclass which requires the execution of the static initializer," as recited in claim 1.

In short, Seshadri fails to disclose or suggest the computer system defined by claim 1, because Seshadri fails to disclose or suggest: (1) a preloader arranged to, (2) determine whether a bytecode makes an active reference to a class which requires an execution of a static initializer, (3) determine if the class has a superclass which requires the execution of the static initializer, and (4) wherein the preloader produces a source file. Because Seshadri fails to teach these features of the computer system of claim 1, Seshadri fails to support an obviousness rejection of claim 1 under 35 U.S.C. § 103(a), taken alone or in combination with Long. Therefore, this rejection should be withdrawn.

Independent claims 6, 7 and 12 incorporate similar features as claim 1. Thus, the cited art, taken alone or in combination, fails to support a rejection of these claims for the same reasons as claim 1.

Dependent claims 2-5, 8-11 and 13-16 incorporate all of the features of the independent claims on which they are based. Therefore, the cited art fails to support any rejection of these claims for the same reasons as the independent claims on which they are based.

### CONCLUSION

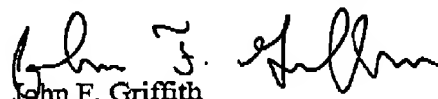
Based on the foregoing, it is respectfully submitted that this application is in condition for allowance. Early notification to that effect is respectfully requested.

If there are any issues remaining after the review of this Response, the Examiner is respectfully requested to contact the undersigned at the telephone number below.

If any fees are due in connection with the filing of this Response (including any fees due for an extension of time), such fees may be charged to Deposit Account No. 500388 (Order No. SUN1P802).

Respectfully submitted,

BEYER WEAVER & THOMAS, LLP

  
John F. Griffith  
Reg. No. 44,137

P.O. Box 778  
Berkeley, CA 94704-0778  
(510) 843-6200

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**